

ATTORNEY'S DOCKET NO. MS1-548US

[illegible]

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

6

7
8
9
10
11
12
13
14
15

16
17
18
19
20
21
22
23
24
25

1 further beneficial to have an improved way to manage remote computers at the
2 user's location without user intervention.

3 The invention described below addresses these disadvantages, restricting
4 data transfers and managing software components of distributed computers.

6 **SUMMARY OF THE INVENTION**

7 Restricting data transfers and managing software components in clusters of
8 server computers located at a co-location facility is described herein.

9 According to one aspect, a controller (referred to as the "BMonitor") is
10 situated on a computer (e.g., each node in a co-location facility). The BMonitor
11 includes a plurality of filters that identify where data can be sent to and/or received
12 from, such as another node in the co-location facility or a client computer coupled
13 to the computer via the Internet. These filters can then be modified, during
14 operation of the computer, by one or more management devices coupled to the
15 computer.

16 According to another aspect, a controller referred to as the "BMonitor"
17 (situated on a computer) manages software components executing on that
18 computer. Requests are received by the BMonitor from external sources and
19 implemented by the BMonitor. Such requests can originate from a management
20 console local to the computer or alternatively remote from the computer.

According to another aspect, a controller referred to as the "BMonitor" (situated on a computer) operates as a trusted third party mediating interaction among multiple management devices. The BMonitor maintains multiple ownership domains, each corresponding to a management device(s) and each having a particular set of rights that identify what types of management functions

they can command the BMonitor to carry out. Only one ownership domain is the top-level domain at any particular time, and the top-level domain has a more expanded set of rights than any of the lower-level domains. The top-level domain can create new ownership domains corresponding to other management device, and can also be removed and the management rights of its corresponding management device revoked at any time by a management device corresponding to a lower-level ownership domain. Each time a change of which ownership domain is the top-level ownership domain occurs, the computer's system memory can be erased so that no confidential information from one ownership domain is made available to devices corresponding to other ownership domains.

According to another aspect, the BMonitor is implemented in a more-privileged level than other software engines executing on the node, preventing other software engines from interfering with restrictions imposed by the BMonitor.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings. The same numbers are used throughout the figures to reference like components and/or features.

Fig. 1 shows a client/server network system and environment such as may be used with certain embodiments of the invention.

Fig. 2 shows a general example of a computer that can be used in accordance with certain embodiments of the invention.

Fig. 3 is a block diagram illustrating an exemplary co-location facility in more detail.

1 be implemented in any of a variety of different manners, including wired and/or
2 wireless communications media.

3 Communication over network 108 can be carried out using any of a wide
4 variety of communications protocols. In one implementation, client computers
5 102 and server computers in clusters 106 can communicate with one another using
6 the Hypertext Transfer Protocol (HTTP), in which web pages are hosted by the
7 server computers and written in a markup language, such as the Hypertext Markup
8 Language (HTML) or the eXtensible Markup Language (XML).

9 Management device 110 operates to manage software components of one or
10 more computing devices located at a location remote from device 110. This
11 management may also include restricting data transfers into and/or out of the
12 computing device being managed. In the illustrated example of Fig. 1,
13 management device 110 can remotely manage any one or more of: a client(s) 102,
14 a server cluster(s) 106, or a server(s) 112. Any of a wide variety of computing
15 devices can be remotely managed, including personal computers (PCs), network
16 PCs, multiprocessor systems, microprocessor-based or programmable consumer
17 electronics, minicomputers, mainframe computers, gaming consoles, Internet
18 appliances, personal digital assistants (PDAs), pocket computers, palm-sized
19 computers, handheld computers, digital cellular phones, etc. Remote management
20 of a computing device is accomplished by communicating commands to the
21 device via network 108, as discussed in more detail below.

22 In the discussion herein, embodiments of the invention are described in the
23 general context of computer-executable instructions, such as program modules,
24 being executed by one or more conventional personal computers. Generally,
25 program modules include routines, programs, objects, components, data structures,

etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that various embodiments of the invention may be practiced with other computer system configurations, including hand-held devices, gaming consoles, Internet appliances, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. In a distributed computer environment, program modules may be located in both local and remote memory storage devices.

Alternatively, embodiments of the invention can be implemented in hardware or a combination of hardware, software, and/or firmware. For example, all or part of the invention can be implemented in one or more application specific integrated circuits (ASICs) or programmable logic devices (PLDs).

Fig. 2 shows a general example of a computer 142 that can be used in accordance with certain embodiments of the invention. Computer 142 is shown as an example of a computer that can perform the functions of a client computer 102 of Fig. 1, a server computer or node in a co-location facility 104 of Fig. 1, a management device 110 of Fig. 1, a server 112 of Fig. 1, or a local or remote management console as discussed in more detail below.

Computer 142 includes one or more processors or processing units 144, a system memory 146, and a bus 148 that couples various system components including the system memory 146 to processors 144. The bus 148 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 150 and random access memory (RAM) 152.

1 A basic input/output system (BIOS) 154, containing the basic routines that help to
2 transfer information between elements within computer 142, such as during start-
3 up, is stored in ROM 150.

4 Computer 142 further includes a hard disk drive 156 for reading from and
5 writing to a hard disk, not shown, connected to bus 148 via a hard disk driver
6 interface 157 (e.g., a SCSI, ATA, or other type of interface); a magnetic disk drive
7 158 for reading from and writing to a removable magnetic disk 160, connected to
8 bus 148 via a magnetic disk drive interface 161; and an optical disk drive 162 for
9 reading from or writing to a removable optical disk 164 such as a CD ROM, DVD,
10 or other optical media, connected to bus 148 via an optical drive interface 165.
11 The drives and their associated computer-readable media provide nonvolatile
12 storage of computer readable instructions, data structures, program modules and
13 other data for computer 142. Although the exemplary environment described
14 herein employs a hard disk, a removable magnetic disk 160 and a removable
15 optical disk 164, it should be appreciated by those skilled in the art that other types
16 of computer readable media which can store data that is accessible by a computer,
17 such as magnetic cassettes, flash memory cards, digital video disks, random access
18 memories (RAMs) read only memories (ROM), and the like, may also be used in
19 the exemplary operating environment.

20 A number of program modules may be stored on the hard disk, magnetic
21 disk 160, optical disk 164, ROM 150, or RAM 152, including an operating system
22 170, one or more application programs 172, other program modules 174, and
23 program data 176. A user may enter commands and information into computer
24 142 through input devices such as keyboard 178 and pointing device 180. Other
25 input devices (not shown) may include a microphone, joystick, game pad, satellite

Computer 142 optionally operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 188. The remote computer 188 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 142, although only a memory storage device 190 has been illustrated in Fig. 2. The logical connections depicted in Fig. 2 include a local area network (LAN) 192 and a wide area network (WAN) 194. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet. In the described embodiment of the invention, remote computer 188 executes an Internet Web browser program (which may optionally be integrated into the operating system 170) such as the "Internet Explorer" Web browser manufactured and distributed by Microsoft Corporation of Redmond, Washington.

When used in a LAN networking environment, computer 142 is connected to the local network 192 through a network interface or adapter 196. When used in a WAN networking environment, computer 142 typically includes a modem 198 or other component for establishing communications over the wide area network 194, such as the Internet. The modem 198, which may be internal or external, is connected to the system bus 148 via an interface (e.g., a serial port interface 168).

1 In a networked environment, program modules depicted relative to the personal
2 computer 142, or portions thereof, may be stored in the remote memory storage
3 device. It is to be appreciated that the network connections shown are exemplary
4 and other means of establishing a communications link between the computers
5 may be used.

6 Generally, the data processors of computer 142 are programmed by means
7 of instructions stored at different times in the various computer-readable storage
8 media of the computer. Programs and operating systems are typically distributed,
9 for example, on floppy disks or CD-ROMs. From there, they are installed or
10 loaded into the secondary memory of a computer. At execution, they are loaded at
11 least partially into the computer's primary electronic memory. The invention
12 described herein includes these and other various types of computer-readable
13 storage media when such media contain instructions or programs for implementing
14 the steps described below in conjunction with a microprocessor or other data
15 processor. The invention also includes the computer itself when programmed
16 according to the methods and techniques described below. Furthermore, certain
17 sub-components of the computer may be programmed to perform the functions
18 and steps described below. The invention includes such sub-components when
19 they are programmed as described. In addition, the invention described herein
20 includes data structures, described below, as embodied on various types of
21 memory media.

22 For purposes of illustration, programs and other executable program
23 components such as the operating system are illustrated herein as discrete blocks,
24 although it is recognized that such programs and components reside at various
25

1 times in different storage components of the computer, and are executed by the
2 data processor(s) of the computer.

3 Fig. 3 is a block diagram illustrating an exemplary co-location facility in
4 more detail. Co-location facility 104 is illustrated including multiple nodes (also
5 referred to as server computers) 210. Co-location facility 104 can include any
6 number of nodes 210, and can easily include an amount of nodes numbering into
7 the thousands.

8 The nodes 210 are grouped together in clusters, referred to as server
9 clusters (or node clusters). For ease of explanation and to avoid cluttering the
10 drawings, only a single cluster 212 is illustrated in Fig. 3. Each server cluster
11 includes nodes 210 that correspond to a particular customer of co-location facility
12 104. The nodes 210 of a server cluster can be physically isolated from the nodes
13 210 of other server clusters. This physical isolation can take different forms, such
14 as separate locked cages or separate rooms at co-location facility 104. Physically
15 isolating server clusters ensures customers of co-location facility 104 that only
16 they can physically access their nodes (other customers cannot).

17 A landlord/tenant relationship (also referred to as a lessor/lessee
18 relationship) can also be established based on the nodes 210. The owner (and/or
19 operator) of co-location facility 104 owns (or otherwise has rights to) the
20 individual nodes 210, and thus can be viewed as a "landlord". The customers of
21 co-location facility 104 lease the nodes 210 from the landlord, and thus can be
22 viewed as a "tenant". The landlord is typically not concerned with what types of
23 data or programs are being stored at the nodes 210 by the tenant, but does impose
24 boundaries on the clusters that prevent nodes 210 from different clusters from
25 communicating with one another, as discussed in more detail below. Additionally,

the nodes 210 provide assurances to the tenant that, although the nodes are only leased to the tenant, the landlord cannot access confidential information stored by the tenant.

Although physically isolated, nodes 210 of different clusters are often physically coupled to the same transport medium (or media) 211 that enables access to network connection(s) 216, and possibly application operations management console 242, discussed in more detail below. This transport medium can be wired or wireless.

As each node 210 can be coupled to a shared transport medium 211, each node 210 is configurable to restrict which other nodes 210 data can be sent to or received from. Given that a number of different nodes 210 may be included in a customer's (also referred to as tenant's) server cluster, the customer may want to be able to pass data between different nodes 210 within the cluster for processing, storage, etc. However, the customer will typically not want data to be passed to other nodes 210 that are not in the server cluster. Configuring each node 210 in the cluster to restrict which other nodes 210 data can be sent to or received from allows a boundary for the server cluster to be established and enforced. Establishment and enforcement of such server cluster boundaries prevents customer data from being erroneously or improperly forwarded to a node that is not part of the cluster.

These initial boundaries established by the landlord prevent communication between nodes 210 of different customers, thereby ensuring that each customer's data can be passed to other nodes 210 of that customer. The customer itself may also further define sub-boundaries within its cluster, establishing sub-clusters of nodes 210 that data cannot be communicated out of (or in to) either to or from

1 other nodes in the cluster. The customer is able to add, modify, remove, etc. such
2 sub-cluster boundaries at will, but only within the boundaries defined by the
3 landlord (that is, the cluster boundaries). Thus, the customer is not able to alter
4 boundaries in a manner that would allow communication to or from a node 210 to
5 extend to another node 210 that is not within the same cluster.

6 Co-location facility 104 supplies reliable power 214 and reliable network
7 connection(s) 216 (e.g., to network 108 of Fig. 1) to each of the nodes 210. Power
8 214 and network connection(s) 216 are shared by all of the nodes 210, although
9 alternatively separate power 214 and network connection(s) 216 may be supplied
10 to nodes 210 or groupings (e.g., clusters) of nodes. Any of a wide variety of
11 conventional mechanisms for supplying reliable power can be used to supply
12 reliable power 214, such as power received from a public utility company along
13 with backup generators in the event of power failures, redundant generators,
14 batteries, fuel cells, or other power storage mechanisms, etc. Similarly, any of a
15 wide variety of conventional mechanisms for supplying a reliable network
16 connection can be used to supply network connection(s) 216, such as redundant
17 connection transport media, different types of connection media, different access
18 points (e.g., different Internet access points, different Internet service providers
19 (ISPs), etc.).

20 In certain embodiments, nodes 210 are leased or sold to customers by the
21 operator or owner of co-location facility 104 along with the space (e.g., locked
22 cages) and service (e.g., access to reliable power 214 and network connection(s)
23 216) at facility 104. In other embodiments, space and service at facility 104 may
24 be leased to customers while one or more nodes are supplied by the customer.
25

1 Management of each node 210 is carried out in a multiple-tiered manner.
2 Fig. 4 is a block diagram illustrating an exemplary multi-tiered management
3 architecture. The multi-tiered architecture includes three tiers: a cluster
4 operations management tier 230, an application operations management tier 232,
5 and an application development tier 234. Cluster operations management tier 230
6 is implemented locally at the same location as the server(s) being managed (e.g., at
7 a co-location facility) and involves managing the hardware operations of the
8 server(s). In the illustrated example, cluster operations management tier 230 is not
9 concerned with what software components are executing on the nodes 210, but
10 only with the continuing operation of the hardware of nodes 210 and establishing
11 any boundaries between clusters of nodes.

12 The application operations management tier 232, on the other hand, is
13 implemented at a remote location other than where the server(s) being managed
14 are located (e.g., other than the co-location facility), but from a client computer
15 that is still communicatively coupled to the server(s). The application operations
16 management tier 232 involves managing the software operations of the server(s)
17 and defining any sub-boundaries within server clusters. The client can be coupled
18 to the server(s) in any of a variety of manners, such as via the Internet or via a
19 dedicated (e.g., dial-up) connection. The client can be coupled continually to the
20 server(s), or alternatively sporadically (e.g., only when needed for management
21 purposes).

22 The application development tier 234 is implemented on another client
23 computer at a location other than the server(s) (e.g., other than at the co-location
24 facility) and involves development of software components or engines for
25 execution on the server(s). Alternatively, current software on a node 210 at co-

1 location facility 104 could be accessed by a remote client to develop additional
2 software components or engines for the node. Although the client at which
3 application development tier 234 is implemented is typically a different client than
4 that at which application operations management tier 232 is implemented, tiers
5 232 and 234 could be implemented (at least in part) on the same client.

6 Although only three tiers are illustrated in Fig. 4, alternatively the multi-
7 tiered architecture could include different numbers of tiers. For example, the
8 application operations management tier may be separated into two tiers, each
9 having different (or overlapping) responsibilities, resulting in a 4-tiered
10 architecture. The management at these tiers may occur from the same place (e.g.,
11 a single application operations management console may be shared), or
12 alternatively from different places (e.g., two different operations management
13 consoles).

14 Returning to Fig. 3, co-location facility 104 includes a cluster operations
15 management console for each server cluster. In the example of Fig. 3, cluster
16 operations management console 240 corresponds to cluster 212 and may be, for
17 example, a management device 110 of Fig. 1. Cluster operations management
18 console 240 implements cluster operations management tier 230 (Fig. 4) for
19 cluster 212 and is responsible for managing the hardware operations of nodes 210
20 in cluster 212. Cluster operations management console 240 monitors the hardware
21 in cluster 212 and attempts to identify hardware failures. Any of a wide variety of
22 hardware failures can be monitored for, such as processor failures, bus failures,
23 memory failures, etc. Hardware operations can be monitored in any of a variety of
24 manners, such as cluster operations management console 240 sending test
25 messages or control signals to the nodes 210 that require the use of particular

1 hardware in order to respond (no response or an incorrect response indicates
2 failure), having messages or control signals that require the use of particular
3 hardware to generate periodically sent by nodes 210 to cluster operations
4 management console 240 (not receiving such a message or control signal within a
5 specified amount of time indicates failure), etc. Alternatively, cluster operations
6 management console 240 may make no attempt to identify what type of hardware
7 failure has occurred, but rather simply that a failure has occurred.

8 Once a hardware failure is detected, cluster operations management console
9 240 acts to correct the failure. The action taken by cluster operations management
10 console 240 can vary based on the hardware as well as the type of failure, and can
11 vary for different server clusters. The corrective action can be notification of an
12 administrator (e.g., a flashing light, an audio alarm, an electronic mail message,
13 calling a cell phone or pager, etc.), or an attempt to physically correct the problem
14 (e.g., reboot the node, activate another backup node to take its place, etc.).

15 Cluster operations management console 240 also establishes cluster
16 boundaries within co-location facility 104. The cluster boundaries established by
17 console 240 prevent nodes 210 in one cluster (e.g., cluster 212) from
18 communicating with nodes in another cluster (e.g., any node not in cluster 212),
19 while at the same time not interfering with the ability of nodes 210 within a cluster
20 from communicating with other nodes within that cluster. These boundaries
21 provide security for the tenants' data, allowing them to know that their data cannot
22 be communicated to other tenants' nodes 210 at facility 104 even though network
23 connection 216 may be shared by the tenants.

24 In the illustrated example, each cluster of co-location facility 104 includes a
25 dedicated cluster operations management console. Alternatively, a single cluster

operations management console may correspond to, and manage hardware operations of, multiple server clusters. According to another alternative, multiple cluster operations management consoles may correspond to, and manage hardware operations of, a single server cluster. Such multiple consoles can manage a single server cluster in a shared manner, or one console may operate as a backup for another console (e.g., providing increased reliability through redundancy, to allow for maintenance, etc.).

An application operations management console 242 is also communicatively coupled to co-location facility 104. Application operations management console 242 may be, for example, a management device 110 of Fig. 1. Application operations management console 242 is located at a location remote from co-location facility 104 (that is, not within co-location facility 104), typically being located at the offices of the customer. A different application operations management console 242 corresponds to each server cluster of co-location facility 104, although alternatively multiple consoles 242 may correspond to a single server cluster, or a single console 242 may correspond to multiple server clusters. Application operations management console 240 implements application operations management tier 232 (Fig. 4) for cluster 212 and is responsible for managing the software operations of nodes 210 in cluster 212 as well as securing sub-boundaries within cluster 212.

Application operations management console 242 monitors the software in cluster 212 and attempts to identify software failures. Any of a wide variety of software failures can be monitored for, such as application processes or threads that are "hung" or otherwise non-responsive, an error in execution of application processes or threads, etc. Software operations can be monitored in any of a variety

of manners (similar to the monitoring of hardware operations discussed above), such as application operations management console 242 sending test messages or control signals to particular processes or threads executing on the nodes 210 that require the use of particular routines in order to respond (no response or an incorrect response indicates failure), having messages or control signals that require the use of particular software routines to generate periodically sent by processes or threads executing on nodes 210 to application operations management console 242 (not receiving such a message or control signal within a specified amount of time indicates failure), etc. Alternatively, application operations management console 242 may make no attempt to identify what type of software failure has occurred, but rather simply that a failure has occurred.

Once a software failure is detected, application operations management console 242 acts to correct the failure. The action taken by application operations management console 242 can vary based on the hardware as well as the type of failure, and can vary for different server clusters. The corrective action can be notification of an administrator (e.g., a flashing light, an audio alarm, an electronic mail message, calling a cell phone or pager, etc.), or an attempt to correct the problem (e.g., reboot the node, re-load the software component or engine image, terminate and re-execute the process, etc.).

Thus, the management of a node 210 is distributed across multiple managers, regardless of the number of other nodes (if any) situated at the same location as the node 210. The multi-tiered management allows the hardware operations management to be separated from the application operations management, allowing two different consoles (each under the control of a different entity) to share the management responsibility for the node.

1 module 260. Network interface 256 provides the interface between node 248 and
2 the network (e.g., network 108 of Fig. 1). Filters 258 identify other nodes 248 in a
3 co-location facility (and/or other sources or targets (e.g., coupled to Internet 108 of
4 Fig. 1) that data can (or alternatively cannot) be sent to and/or received from. The
5 nodes or other sources/targets can be identified in any of a wide variety of
6 manners, such as by network address (e.g., Internet Protocol (IP) address), some
7 other globally unique identifier, a locally unique identifier (e.g., a numbering
8 scheme proprietary or local to co-location facility 104), etc.

9 Filters 258 can fully restrict access to a node (e.g., no data can be received
10 from or sent to the node), or partially restrict access to a node. Partial access
11 restriction can take different forms. For example, a node may be restricted so that
12 data can be received from the node but not sent to the node (or vice versa). By
13 way of another example, a node may be restricted so that only certain types of data
14 (e.g., communications in accordance with certain protocols, such as HTTP) can be
15 received from and/or sent to the node. Filtering based on particular types of data
16 can be implemented in different manners, such as by communicating data in
17 packets with header information that indicate the type of data included in the
18 packet.

19 Filters 258 can be added by one or more management devices 110 of Fig. 1
20 or either of application operations management console 242 or cluster operations
21 management console 240 of Fig. 3. In the illustrated example, filters added by
22 cluster operations management console 240 (to establish cluster boundaries)
23 restrict full access to nodes (e.g., any access to another node can be prevented)
24 whereas filters added by application operations management console 242 (to
25

passing through to the target. Controller 254 may return an indication to the source of the data that the data cannot be passed to the target, or may simply ignore or discard the data.

The application of filters 258 to the data by controller 254 allows the boundary restrictions of a server cluster (Fig. 3) to be imposed. Filters 258 can be programmed (e.g., by application operations management console 242 of Fig. 3) with the node addresses of all the nodes within the server cluster (e.g., cluster 212). Controller 254 then prevents data received from any node not within the server cluster from being passed through to an engine 252, and similarly prevents any data being sent to a node other than one within the server cluster from being sent. Similarly, data received from Internet 108 (Fig. 1) can identify a target node 248 (e.g., by IP address), so that controller 254 of any node other than the target node will prevent the data from being passed through to an engine 252. Furthermore, as filters 258 can be readily modified by cluster operations management console 240, server cluster boundaries can be easily changed to accommodate changes in the server cluster (e.g., addition of nodes to and/or removal of nodes from the server cluster).

BMCP module 260 implements the Distributed Host Control Protocol (DHCP), allowing BMonitor 250 (and thus node 248) to obtain an IP address from a DHCP server (e.g., cluster operations management console 240 of Fig. 3). During an initialization process for node 248, BMCP module 260 requests an IP address from the DHCP server, which in turn provides the IP address to module 260. Additional information regarding DHCP is available from Microsoft Corporation of Redmond, Washington.

Software engines 252 include any of a wide variety of conventional software components. Examples of engines 252 include an operating system (e.g., Windows NT®), a load balancing server component (e.g., to balance the processing load of multiple nodes 248), a caching server component (e.g., to cache data and/or instructions from another node 248 or received via the Internet), a storage manager component (e.g., to manage storage of data from another node 248 or received via the Internet), etc. In one implementation, each of the engines 252 is a protocol-based engine, communicating with BMonitor 250 and other engines 252 via messages and/or function calls without requiring the engines 252 and BMonitor 250 to be written using the same programming language.

Controller 254, in conjunction with loader 264, is responsible for controlling the execution of engines 252. This control can take different forms, including beginning or initiating execution of an engine 252, terminating execution of an engine 252, re-loading an image of an engine 252 from a storage device, debugging execution of an engine 252, etc. Controller 254 receives instructions from application operations management console 242 of Fig. 3 or a management device(s) 110 of Fig. 1 regarding which of these control actions to take and when to take them. In the event that execution of an engine 252 is to be initiated (including re-starting an engine whose execution was recently terminated), controller 254 communicates with loader 264 to load an image of the engine 252 from a storage device (e.g., device 262, ROM, etc.) into the memory (e.g., RAM) of node 248. Loader 264 operates in a conventional manner to copy the image of the engine from the storage device into memory and initialize any necessary operating system parameters to allow execution of the engine 252.

1 Thus, the control of engines 252 is actually managed by a remote device, not
2 locally at the same location as the node 248 being managed.

3 Controller 254 also provides an interface via which application operations
4 management console 242 of Fig. 3 or a management device(s) 110 of Fig. 1 can
5 identify filters to add (and/or remove) from filter set 258.

6 Controller 254 also includes an interface via which cluster operations
7 management console 240 of Fig. 3 can communicate commands to controller 254.
8 Different types of hardware operation oriented commands can be communicated to
9 controller 254 by cluster operations management console 240, such as re-booting
10 the node, shutting down the node, placing the node in a low-power state (e.g., in a
11 suspend or standby state), changing cluster boundaries, changing encryption keys
12 (if any), etc.

13 Controller 254 further optionally provides encryption support for BMonitor
14 250, allowing data to be stored securely on mass storage device 262 (e.g., a
15 magnetic disk, an optical disk, etc.) and secure communications to occur between
16 node 248 and an operations management console (e.g., console 240 or 242 of Fig.
17 3) or other management device (e.g., management device 110 of Fig. 1).
18 Controller 254 maintains multiple encryption keys 259, which can include a
19 variety of different keys such as symmetric keys (secret keys used in secret key
20 cryptography), public/private key pairs (for public key cryptography), etc. to be
21 used in encrypting and/or decrypting data.

22 BMonitor 250 makes use of public key cryptography to provide secure
23 communications between node 248 and the management consoles (e.g., consoles
24 240 or 242 of Fig. 3) or other management devices (e.g., management device(s)
25 110 of Fig. 1). Public key cryptography is based on a key pair, including both a

1 public key and a private key, and an encryption algorithm. The encryption
2 algorithm can encrypt data based on the public key such that it cannot be
3 decrypted efficiently without the private key. Thus, communications from the
4 public-key holder can be encrypted using the public key, allowing only the
5 private-key holder to decrypt the communications. Any of a variety of public key
6 cryptography techniques may be used, such as the well-known RSA (Rivest,
7 Shamir, and Adelman) encryption technique. For a basic introduction of
8 cryptography, the reader is directed to a text written by Bruce Schneier and
9 entitled "Applied Cryptography: Protocols, Algorithms, and Source Code in C,"
10 published by John Wiley & Sons with copyright 1994 (or second edition with
11 copyright 1996).

12 BMonitor 250 is initialized to include a public/private key pair for both the
13 landlord and the tenant. These key pairs can be generated by BMonitor 250, or
14 alternatively by some other component and stored within BMonitor 250 (with that
15 other component being trusted to destroy its knowledge of the key pair). As used
16 herein, U refers to a public key and R refers to a private key. The public/private
17 key pair for the landlord is referred to as (U_L, R_L) , and the public/private key pair
18 for the tenant is referred to as (U_T, R_T) . BMonitor 250 makes the public keys U_L
19 and U_T available to the landlord, but keeps the private keys R_L and R_T secret. In
20 the illustrated example, BMonitor 250 never divulges the private keys R_L and R_T ,
21 so both the landlord and the tenant can be assured that no entity other than the
22 BMonitor 250 can decrypt information that they encrypt using their public keys
23 (e.g., via cluster operations management console 240 and application operations
24 management console 242 of Fig. 3, respectively).

25

1 250 keeps the disk key secure, using it only to encrypt data node stored on mass
2 storage device 262 and decrypt data node retrieved from mass storage device 262
3 (thus there is no need for any other entities, including any management device, to
4 have knowledge of the disk key).

5 Use of the disk key ensures that data stored on mass storage device 262 can
6 only be decrypted by the node that encrypted it, and not any other node or device.
7 Thus, for example, if mass storage device 262 were to be removed and attempts
8 made to read the data on device 262, such attempts would be unsuccessful.
9 BMonitor 250 uses the disk key to encrypt data to be stored on mass storage
10 device 262 regardless of the source of the data. For example, the data may come
11 from a client device (e.g., client 102 of Fig. 1) used by a customer of the tenant,
12 from a management device (e.g., a device 110 of Fig. 1 or a console 240 or 242 of
13 Fig. 3), etc.

14 In one implementation, the disk key is generated by combining the storage
15 keys corresponding to each management device. The storage keys can be
16 combined in a variety of different manners, and in one implementation are
17 combined by using one of the keys to encrypt the other key, with the resultant
18 value being encrypted by another one of the keys, etc.

19 Additionally, BMonitor 250 operates as a trusted third party mediating
20 interaction among multiple mutually distrustful management agents that share
21 responsibility for managing node 248. For example, the landlord and tenant for
22 node 248 do not typically fully trust one another. BMonitor 250 thus operates as a
23 trusted third party, allowing the lessor and lessee of node 248 to trust that
24 information made available to BMonitor 250 by a particular entity or agent is
25 accessible only to that entity or agent, and no other (e.g., confidential information

1 (removed from) stack 286 along with any other higher-level ownership domains.
 2 For example, if the owner of node 248 (ownership domain 280) were to revoke the
 3 rights of ownership domain 282, then ownership domains 282 and 284 would be
 4 popped from ownership domain stack 286.

5 Each ownership domain has a corresponding set of rights. In the illustrated
 6 example, the top-level ownership domain has one set of rights that include: (1) the
 7 right to push new ownership domains on the ownership domain stack; (2) the right
 8 to access any system memory in the node; (3) the right to access any mass storage
 9 devices in or coupled to the node; (4) the right to modify (add, remove, or change)
 10 packet filters at the node; (5) the right to start execution of software engines on the
 11 node (e.g., engines 252 of Fig. 5); (6) the right to stop execution of software
 12 engines on the node, including resetting the node; (7) the right to debug software
 13 engines on the node; (8) the right to change its own authentication credentials
 14 (e.g., its public key or ID); (9) the right to modify its own storage key; (10) the
 15 right to subscribe to events engine events, machine events, and/or packet filter
 16 events (e.g., notify a management console or other device when one of these
 17 events occurs). Additionally, each of the lower-level ownership domains has
 18 another set of rights that include: (1) the right to pop an existing ownership
 19 domain(s); (2) the right to modify (add, remove, or change) packet filters at the
 20 node; (3) the right to change its own authentication credentials (e.g., public key or
 21 ID); and (4) the right to subscribe to machine events and/or packet filter events.
 22 Alternatively, some of these rights may not be included (e.g., depending on the
 23 situation, the right to debug software engines on the node may not be needed), or
 24 other rights may be included (e.g., the top-level node may include the right to pop
 25 itself off the ownership domain stack).

1 Ownership domains can be added to and removed from ownership domain
2 stack 286 numerous times during operation. Which ownership domains are
3 removed and/or added varies based on the activities being performed. By way of
4 example, if the owner of node 248 (corresponding to root ownership domain 280)
5 desires to perform some operation on node 248, all higher-level ownership
6 domains 282 – 284 are revoked, the desired operation is performed (ownership
7 domain 280 is now the top-level domain, so the expanded set of rights are
8 available), and then new ownership domains can be created and added to
9 ownership domain stack 286 (e.g., so that the management agent previously
10 corresponding to the top-level ownership domain is returned to its previous
11 position).

12 BMonitor 250 checks, for each request received from an entity
13 corresponding to one of the ownership domains (e.g., a management console
14 controlled by the entity), what rights the ownership domain has. If the ownership
15 domain has the requisite rights for the request to be implemented, then BMonitor
16 250 carries out the request. However, if the ownership domain does not have the
17 requisite set of rights, then the request is not carried out (e.g., an indication that the
18 request cannot be carried out can be returned to the requestor, or alternatively the
19 request can simply be ignored).

20 In the illustrated example, each ownership domain includes an identifier
21 (ID), a public key, and a storage key. The identifier serves as a unique identifier of
22 the ownership domain, the public key is used to send secure communications to a
23 management device corresponding to the ownership domain, and the storage key
24 is used (at least in part) to encrypt information stored on mass storage devices. An
25 additional private key may also be included for each ownership domain for the

management device corresponding to the ownership domain to send secure communications to the BMonitor. When the root ownership domain 280 is created, it is initialized (e.g., by BMonitor 250) with its ID and public key. The root ownership domain 280 may also be initialized to include the storage key (and a private key), or alternatively it may be added later (e.g., generated by BMonitor 250, communicated to BMonitor 250 from a management console, etc.). Similarly, each time a new ownership domain is created, the ownership domain that creates the new ownership domain communicates an ID and public key to BMonitor 250 for the new ownership domain. A storage key (and a private key) may also be created for the new ownership domain when the new ownership domain is created, or alternatively at a later time.

12 BMonitor 250 authenticates a management device(s) corresponding to each
13 of the ownership domains. BMonitor does not accept any commands from a
14 management device until it is authenticated, and only reveals confidential
15 information (e.g., encryption keys) for a particular ownership domain to a
16 management device(s) that can authenticate itself as corresponding to that
17 ownership domain. This authentication process can occur multiple times during
18 operation of the node, allowing the management devices for one or more
19 ownership domains to change over time. The authentication of management
20 devices can occur in a variety of different manners. In one implementation, when
21 a management device requests a connection to BMonitor 250 and asserts that it
22 corresponds to a particular ownership domain, BMonitor 250 generates a token
23 (e.g., a random number), encrypts the token with the public key of the ownership
24 domain, and then sends the encrypted token to the requesting management device.
25 Upon receipt of the encrypted token, the management device decrypts the token

1 using its private key, and then returns the decrypted token to BMonitor 250. If the
 2 returned token matches the token that BMonitor 250 generated, then the
 3 authenticity of the management device is verified (because only the management
 4 device with the corresponding private key would be able to decrypt the token). An
 5 analogous process can be used for BMonitor 250 to authenticate itself to the
 6 management device.

7 Once authenticated, the management device can communicate requests to
 8 BMonitor 250 and have any of those requests carried out (assuming it has the
 9 rights to do so). Although not required, it is typically prudent for a management
 10 console, upon initially authenticating itself to BMonitor 250, to change its public
 11 key/private key pair.

12 When a new ownership domain is created, the management device that is
 13 creating the new ownership domain can optionally terminate any executing
 14 engines 252 and erase any system memory and mass storage devices. This
 15 provides an added level of security, on top of the encryption, to ensure that one
 16 management device does not have access to information stored on the hardware by
 17 another management device. Additionally, each time an ownership domain is
 18 popped from the stack, BMonitor 250 terminated any executing engines 252,
 19 erases the system memory, and also erases the storage key for that ownership
 20 domain. Thus, any information stored by that ownership domain cannot be
 21 accessed by the remaining ownership domains – the memory has been erased so
 22 there is no data in memory, and without the storage key information on the mass
 23 storage device cannot be decrypted. BMonitor 250 may alternatively erase the
 24 mass storage device too. However, by simply erasing the key and leaving the data
 25

1 encrypted, BMonitor 250 allows the data to be recovered if the popped ownership
2 domain is re-created (and uses the same storage key).

3 Fig. 7 is a flow diagram illustrating the general operation of BMonitor 250
4 in accordance with certain embodiments of the invention. Initially, BMonitor 250
5 monitors the inputs it receives (block 290). These inputs can be from a variety of
6 different sources, such as another node 248, a client computer via network
7 connection 216 (Fig. 3), client operations management console 240, application
8 operations management console 242, an engine 252, a management device 110
9 (Fig. 1), etc.

10 If the received request is a control request (e.g., from one of consoles 240
11 or 242 of Fig. 1, or a management device(s) 110 of Fig. 1), then a check is made
12 (based on the top-level ownership domain) as to whether the requesting device has
13 the necessary rights for the request (block 292). If the requesting device does not
14 have the necessary rights, then BMonitor 250 returns to monitoring inputs (block
15 290) without implementing the request. However, if the requesting device has the
16 necessary rights, then the request is implemented (block 294), and BMonitor 250
17 continues to monitor the inputs it receives (block 290). However, if the received
18 request is a data request (e.g., inbound from another node 248 or a client computer
19 via network connection 216, outbound from an engine 252, etc.), then BMonitor
20 250 either accepts or rejects the request (act 296), and continues to monitor the
21 inputs it receives (block 290). Whether BMonitor 250 accepts a request is
22 dependent on the filters 258 (Fig. 5), as discussed above.

23 Fig. 8 is a flowchart illustrating an exemplary process for handling
24 outbound data requests in accordance with certain embodiments of the invention.
25 The process of Fig. 8 is implemented by BMonitor 250 of Fig. 5, and may be

1 performed in software. The process of Fig. 9 is discussed with additional
2 reference to components in Fig. 5.

3 Initially, the inbound data request is received (act 310). Controller 254
4 compares the request to inbound request restrictions (act 312). This comparison is
5 accomplished by accessing information corresponding to the data to the inbound
6 request restrictions maintained by filters 258. This comparison allows BMonitor
7 250 to determine whether it is permissible for any of software engines 252 to
8 receive the data request (act 314). For example, if filters 258 indicate which
9 sources data can be received from, then it is permissible for an engine 252 to
10 receive the data request only if the source of the data is identified in filters 258.

11 If it is permissible to receive the inbound data request, then BMonitor 250
12 forwards the request to the targeted engine(s) 252 (act 316). However, if it is not
13 permissible to receive the inbound data request from the source, then BMonitor
14 250 rejects the request (act 318). BMonitor 250 may optionally transmit an
15 indication to the source of the request that it was rejected, or alternatively may
16 simply drop the request.

17 18 **Conclusion**

19 Although the description above uses language that is specific to structural
20 features and/or methodological acts, it is to be understood that the invention
21 defined in the appended claims is not limited to the specific features or acts
22 described. Rather, the specific features and acts are disclosed as exemplary forms
23 of implementing the invention.

24

25